포팅 매뉴얼

SSAFY 7기 구미캠퍼스

특화 프로젝트 - 빅데이터(분산)

담당 컨설턴트 : 강시몬

D110 이상민(팀장), 김주영, 박진경, 이기종, 이재영

신혼부부를 위한 주거지 추천 서비스

살만할지도

목차

1. 기술 스택

- 이슈관리: Jira
- 형상관리: Gitlab
- 커뮤니케이션: Mattermost, Webex, Notion
- 개발환경
    - OS: Window 10
    - IDE
        - IntelliJ IEDA
        - Visual Studio Code: 1.67
        - UI/UX: Figma
    - Database
        - Server: AWS RDS
        - DBMS: MySQL 8.0.28
        - DBMS: MongoDB 4.2
    - Server: AWS EC2
        - OS: Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)
        - SSH: MobaXterm
    - CI/CD: Jenkins, Docker, Nginx
- 상세 기술
    - Backend
        - local
            - JDK: 11
            - Spring Boot: 2.7.3
            - docker : 20.10.12
        - server (ec2)
            - docker : 20.10.12

        - jenkins : 2.346.2
        - nginx : 1.23.1
        - mariaDB : 10.8.3
        - mongoDB : 4.2
        - apache spark : 2.12
    - Frontend
        - HTML5, CSS3, JavaScript(ES6)
        - React.js : 18.0
        - Node.js : 16.17.0 LTS
        - npm : 8.15.0
        - sass : 1.54.9
        - Material-UI : 5.10.7
        - react-redux : 8.0.2
        - react-router-dom : 6.4.0
        - axios : 0.27.2
        - redux : 4.2.0
        - mapbox-gl : 1.13.0

2. 빌드 상세 내용

2.1. JVM, 웹서버, WAS 종류 및 설정값, 버전 등

- JVM : openjdk:11-jdk (back/demo/Dockerfile 참조)
- IntelliJ : 2022.2
- node.js : 16.16.0
- docker : 20.10.12
- spark : 2.12
- webserver : nginx 1.23.1 (Docker)
  - docker를 통해 container로 띄움
  - `docker run -itd --name webserver -v /home/ubuntu/docker-image/ssl:/ssl -p 443:443 -p 80:80 nginx`

```
ubuntu@ip-172-26-1-0:~/docker-volume/ssl$ ls
fullchain.pem  key.p12  privkey.pem
```

  - nginx container 내부 /etc/nginx/conf.d/deafault.conf 내부 변경

```
server {
    listen       443;
    server_name  j7d110.p.ssafy.io;

    #access_log  /var/log/nginx/host.access.log  main;

    ssl    on;
    ssl_certificate    /ssl/fullchain.pem;
    ssl_certificate_key /ssl/privkey.pem;


    location /api {
        #root   /usr/share/nginx/html;
        #index  index.html index.htm;
        proxy_pass https://j7d110.p.ssafy.io:9090;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
    }

    location / {
        root /usr/share/nginx/html/build;
        index index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}
```

- database : mariadb 10.8.3 (Docker)
  - docker를 통해 container로 띄움
  - `docker run -itd --name mariadb -e MYSQL_ROOT_PASSWORD=salman -p 3306:3306 mariadb`
  - mariadb container 내부 /etc/mysql/my.cnf 내용

```
[mariadbd]
skip-host-cache
skip-name-resolve

!includedir /etc/mysql/mariadb.conf.d/
!includedir /etc/mysql/conf.d/

lower_case_table_names = 1
collation-server = utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server = utf8
```

  - ‘salman’ database 생성

```
MariaDB [salman]> show tables;
+-----------------+
| Tables_in_salman |
+-----------------+
| gu_gun_code      |
| news             |
| si_do_code       |
+-----------------+
3 rows in set (0.000 sec)
```

- database : mongodb 4.2 (Docker)

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
openapi  0.001GB
> use openapi
switched to db openapi
> show collections
academy
animalhospital
animalsalon
caraccident
categories
categories_like
categories_search
childsafety
concerthall
crime
drugstore
entertainment
evc
facilitiesforthedisabled
femalesafety
hospital
jeonse
library
mart
park
school
shelter
sportsfacilities
theater
trading
```

2.2. 빌드 주요 내용

- Dockerfile

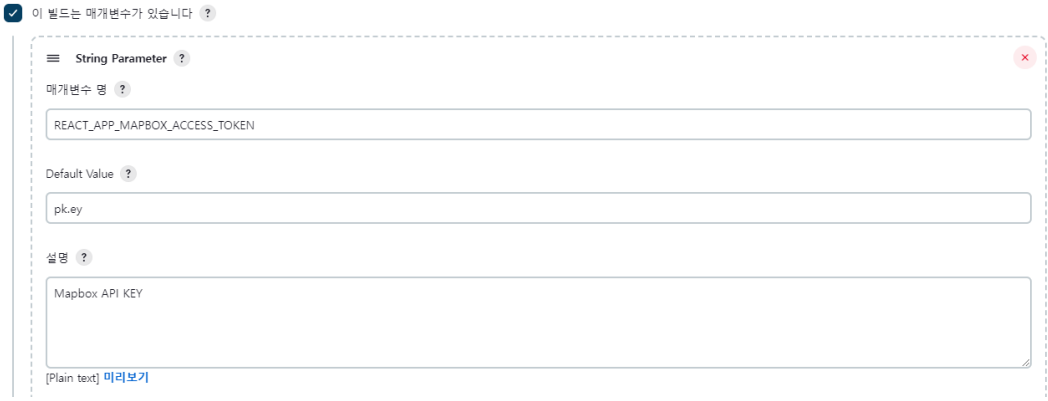
```
FROM openjdk:11-jdk AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM openjdk:11-jdk
COPY --from=builder build/libs/*.jar app.jar
COPY --from=builder ./src ./src
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=deploy","/app.jar"]
```

  - gradle을 통해 빌드한다.
  - 빌드한 파일을 openjdk:11-jdk 환경에서 `java -jar -Dsrpong.profiles.active=deploy boot/app.jar`를 통해 실행시킨다.
  - deploy → application-deploy.properties 파일 사용

- deploy.sh

```
docker build -t salmanhaljido_server .

DOCKER_APP_NAME=salmanhaljido_server

EXIST_BLUE=$(docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml ps | grep Up)

if [ -z "$EXIST_BLUE" ]; then
  echo "BLUE UP"
  docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml up -d
  sleep 20
  START_PORT=9090
  TERMINATE_PORT=9091
  docker exec webserver sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/default.conf
  echo "nginx reload..."
  docker exec webserver service nginx reload
  sleep 5
  docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml down
else
  echo "GREEN UP"
  docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml up -d
  sleep 20
  START_PORT=9091
  TERMINATE_PORT=9090
  docker exec webserver sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/default.conf
  echo "nginx reload..."
  docker exec webserver service nginx reload
  sleep 5
  docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml down
fi
```

- Dockerfile을 통해 salmanhaljido_server라는 이름의 이미지로 빌드한다.
- docker-compose를 이용해 salmanhaljido_server-blue라는 container가 돌아가는지 확인하고, 존재한다면 docker-compose를 이용해 salmanhaljido_server-green container를 올리고 nginx container 내부의 9090과 9091을 바꾼 후, nginx를 reload 해준다.
- 만약 salmanhaljido_server-blue가 없다면 반대로 진행해준다.
- 프론트 빌드를 위해 jenkins에서 매개변수 `REACT_APP_MAPBOX_ACCESS_TOKEN`을 생성하고 Mapbox API의 access token으로 설정한다.



2.3. 배포 시 특이사항

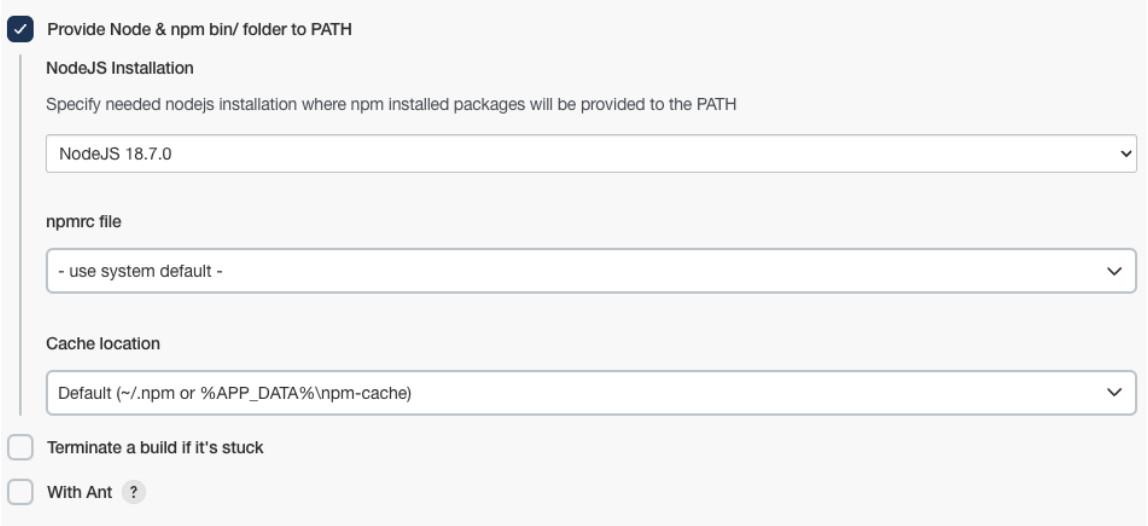
- 배포시 deploy.sh를 실행시키면 Dockerfile과 docker-compose 파일을 사용하여 알아서 docker image를 만들고 docker-compose 파일을 이용하여 container를 띄운다.
  - 배포시 jenkins에서 deploy.sh를 실행시키기 때문에 jenkins는 사용자 권한이 있어야 한다.

  - jenkins또한 docker로 띄웠기 때문에 jenkins container 내부에 docker, docker-compose 설치해주고 사용자 권한을 줘야 한다.

- 해당 container는 9090:8080 or 9091:8080으로 포트 설정이 되어있으며 nginx(Docker)를 reverse_proxy로 사용하여 https://j7d110.p.ssafy.io/api url로 들어온 모든 요청을 해당 container로 돌린다.
- 아무런 table이 없는 DB에 초기화 데이터를 사용하기 위해선 application-deploy.properties 내부 설정을 변경해준다. (1회)

```
spring.config.activate.on-profile=deploy

server.servlet.context-path=/api

spring.datasource.url=jdbc:mariadb://j7d110.p.ssafy.io:3306/salman
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=salman

spring.data.mongodb.uri=mongodb://admin:salmand110@j7d110.p.ssafy.io/openapi?authS
ource=admin

spring.jpa.hibernate.ddl-auto=update

filepath=src/main/resources/data/

server.ssl.enabled=true
server.ssl.key-store=classpath:key.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=salmanhaljido
```

  - 1회만 해당 properties로 빌드 후 이후에는 원래대로 빌드해준다.
- 프론트 배포시 jenkins에서 제공해주는 nodejs 플러그인을 사용해서 빌드 후 nginx container 내부에 docker cp로 옮겨준다.



2.4. 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일

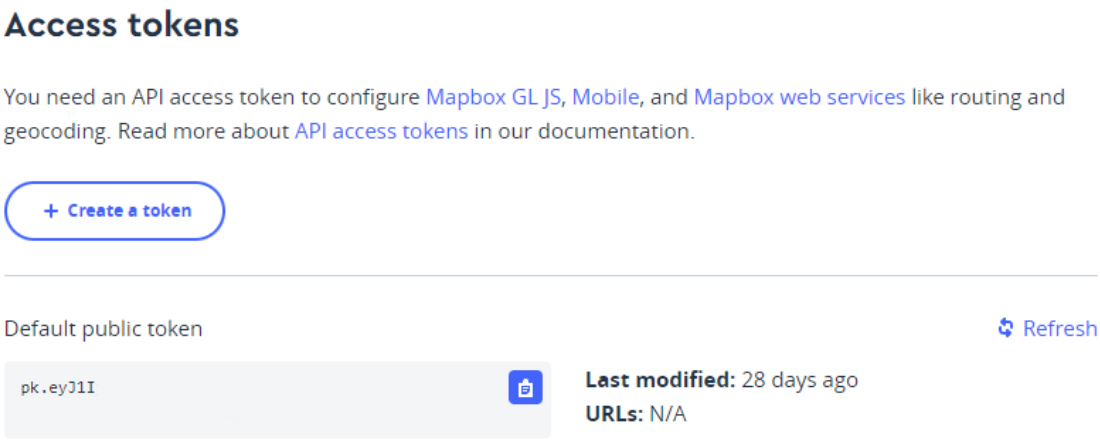
- back/demo/src/main/resources/application-deploy.properties

3. 외부 서비스

3.1. Mapbox API

자유롭고 상세한 커스텀 서비스를 지원하는 지도 API로, 사용을 위해 access token 발급이 필요하다.

1. Mapbox(https://www.mapbox.com/) 접속 및 가입
2. account 접속하여 Default public token 복사



3. 발급 받은 token은 배포 시 jenkins에서 매개변수로 설정하거나 로컬 실행 시 환경변수로 설정한다.
   (2.2 참고)